Project Title: E-Learning Platform

Group Members:

1. Sabahat Jahangir	*22F-3617*
2. Ishrat Fatima	*22F-3616*

Project Deliverables

Table of contents

Introduction

An e-learning platform aims to provide a flexible, accessible, and interactive learning environment. This platform will cater to a diverse range of students, offering a variety of courses and learning materials.

Problem Statement

Traditional education systems are constrained by geographical, physical, and time limitations. As students and instructors increasingly rely on technology for communication and learning, a flexible, scalable, and accessible e-learning platform is needed. The current systems lack user-friendly interfaces, efficient communication tools, and integration with modern content formats (e.g., video, interactive quizzes). This project aims to design and implement an e-learning platform that addresses these gaps.

Scope

The scope of this e-learning platform includes:

- **Course Management:**

  Creating, organizing, and delivering courses with various content formats (e.g., videos, quizzes, discussions).

- **Student Management:**

  Managing student profiles, tracking progress, and issuing certificates.

- **Communication:**

  Facilitating communication between students and instructors (e.g., forums, messaging).

- **Assessment:**

  Creating and administering assessments (e.g., quizzes, assignments).

Functional Requirements

- **User Registration and Login:**

  The system shall allow the users to create accounts and log in to the platform.

- **Course Enrollment:**

  The system shall enable students to browse and enroll in available courses.

- **Course Content Delivery:**

  The system shall be able to present course content in various formats (e.g., videos, text, quizzes).

- **Assessment Creation and Administration:**

  The system shall allow the instructors to create and administer assessments.

- **Communication Features:**

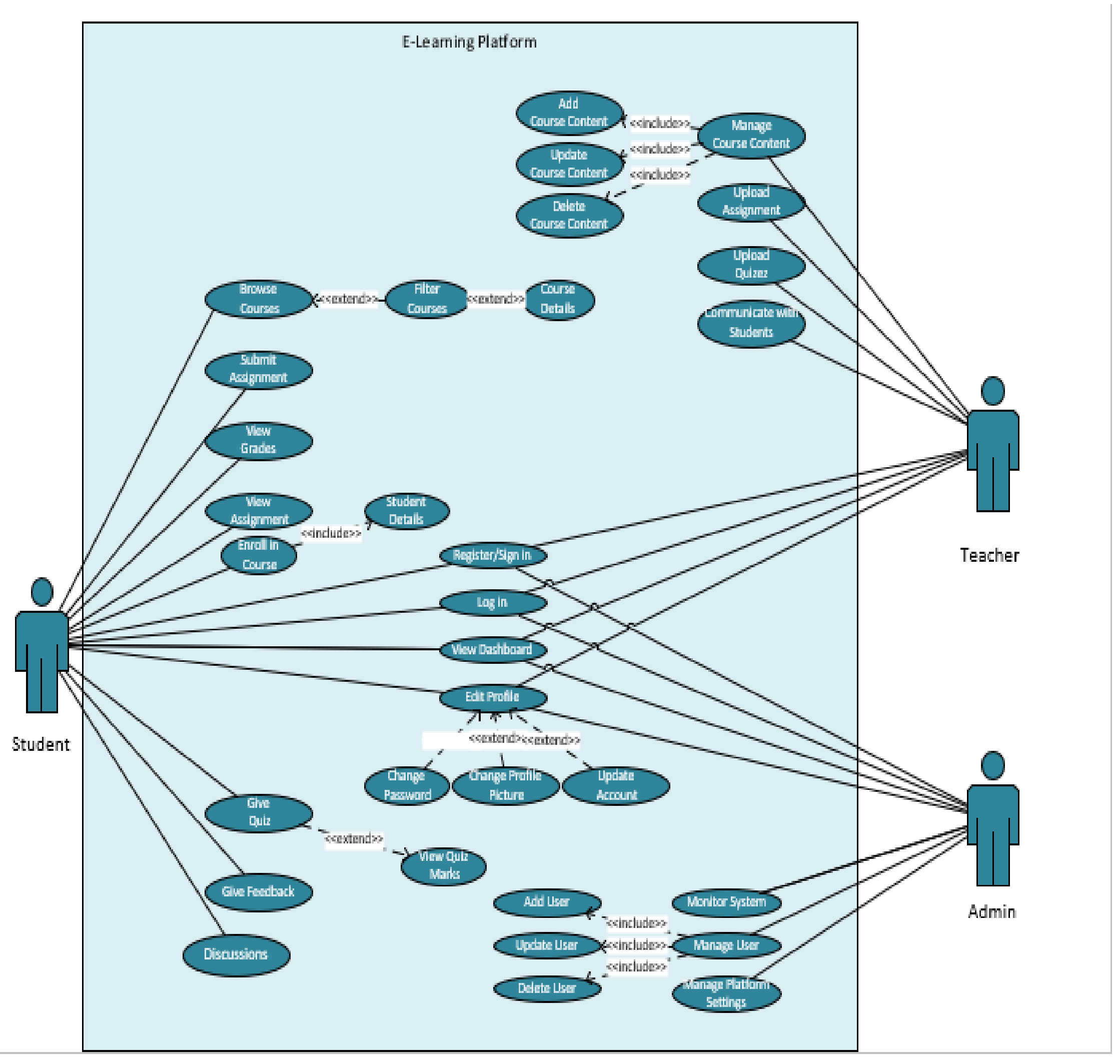
  The system shall provide features for communication between students and instructors (e.g., forums, messaging).

Non-Functional Requirements:

1. **Performance**
   - The system shall support at least 10,000 concurrent users without significant performance degradation.
   - The system shall load product pages within 2 seconds under normal usage conditions.
2. **Scalability**
   - The system shall be scalable to support an increasing number of users over time.
   - The system shall support adding new features or services without affecting the existing functionality.
3. **Security**
   - The system shall use SSL encryption to secure all sensitive data transmitted between users and the server.
   - The system shall implement role-based access control to restrict administrative functions to authorized personnel.

4. **Usability**
    - The system shall have an interface that is intuitive, easy to navigate, and allows users to accomplish tasks efficiently with minimal effort.
    - The system shall be accessible from mobile devices, supporting responsive design for both Android and iOS platforms.
5. **Availability**
    - The system shall have a 99.9% uptime to ensure that users can access the platform at all times.
    - The system shall provide automatic backup and recovery in case of system failures to prevent data loss.
6. **Maintainability**
    - The system shall be designed with modularity in mind to allow for easy maintenance, updates, and bug fixes without extensive downtime.
7. **Reliability**
    - The system shall be able to recover from unexpected shutdowns or crashes without data corruption.
8. **Response Time**
    - The system shall respond to 90% of user requests within 1 second under normal operating conditions.

Use Case Diagram



Use Case Description and Extended Use Cases

Login	
Actor:	Student, Instructor, Admin
Description:	Users log into the platform using valid credentials (username/password).
Preconditions:	User must have an existing account.
Flow of Events:	
User Action	System Response
1. The user enters their credentials (username and password) and clicks “login”.	2. The system validates the credentials. 2.1 If valid, the user is authenticated and redirected to their respective dashboard (student, instructor, or admin). 3. The system logs the successful login.
Postconditions: User is authenticated and redirected to the appropriate dashboard.	
Alternate Flow: □ **Invalid Credentials**: • The system displays an error message ("Invalid username or password. Please try again.") and prompts the user to retry. □ **Forgot Password**: • User clicks "Forgot Password". • The system prompts the user to enter their email to reset the password, sending a reset link to the registered email.	

Enroll in Course:	
Actor:	Student
Description:	Student selects a course and enrolls to gain access to its content.
Preconditions:	Student must be logged in and the course must be available.
Flow of Events:	
User Action	**System Response**
1. The student selects a course from the course catalog and clicks "Enroll".	1. The system verifies that the course is available and that the student is eligible. 2. The system updates the database to reflect the student's enrollment. 3. The student is granted access to course materials and redirected to the course page.
Postconditions: Student is enrolled and can access course materials.	
Alternative Flow: **Course Full**: - The system checks the course capacity and displays an error message if the course is full: "Enrollment closed. Course capacity reached." **Prerequisite Not Met**: - The system checks prerequisites and displays an error message if the student does not meet the requirements: "You do not meet the prerequisites for this course."	

Access Course Content	
Actor:	Student
Description:	Student views or downloads course materials such as videos, documents, and other resources.
Preconditions:	Student must be enrolled in the course.
Flow of Events:	
User Action	**System Response**
1. The student navigates to a specific course and selects a content item (e.g., a video, document, or quiz).	2. The system retrieves the content from the server. 3. The system displays the content on the student’s screen (or initiates download in the case of files).
Postconditions: Student successfully accesses the course materials.	

Submit Assignment:	
Actor:	Student
Description:	Student submits completed assignments through the platform.
Preconditions:	The assignment submission feature is enabled, and the student is enrolled in the course.
Flow of Events:	
User Action	**System Response**
1. The student selects an assignment, uploads their completed work, and clicks "Submit".	2. The system verifies the file format and size. 3. If valid, the system stores the submission in the database and confirms successful submission. 4. The system logs the date and time of the submission and notifies the instructor.
Postconditions: Assignment is uploaded and submitted for grading.	

Alternate Flow:

□ **Invalid File Format/Size**:

- The system checks the file and displays an error message if the file format or size is not supported: "Invalid file format or size. Please upload a valid file."

□ **Late Submission**:

- If the submission is after the deadline, the system may display a warning or prevent the upload depending on course policy: "The assignment submission is late. Please contact the instructor."

<table>
<tr><th colspan="2">Participate in Discussions</th></tr>
<tr><td><b>Actor:</b></td><td>Student</td></tr>
<tr><td><b>Description:</b></td><td>Students engage in discussions with peers and instructors via forums or chat.</td></tr>
<tr><td><b>Preconditions:</b></td><td>Discussion forums or chat should be enabled.</td></tr>
<tr><td colspan="2"><b>Flow of Events:</b></td></tr>
<tr><td><b>User Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1. The student selects a discussion forum or chatroom, types a message, and clicks "Post" or "Send".</td><td>2. The system updates the discussion board for all users currently viewing the thread.</td></tr>
<tr><td colspan="2"><b>Post conditions:</b> Student can post comments and replies in discussion.</td></tr>
</table>

Give Quiz

Actor:	Student
Description:	The student participates in and completes a quiz created by the instructor. The quiz may consist of various question types such as multiple-choice, short answer, or essay questions. The quiz is typically time-limited and must be submitted before the deadline.
Preconditions:	• The student is logged into the platform. • The student is enrolled in the course. • The quiz has been published by the instructor and is within the availability period. • The student has not previously completed the quiz.

Flow of Events:

User Action	**System Response**
1. The student logs in and navigates to the course page where the quiz is listed.	2. The system displays the available quizzes with their due dates and time limits.
3. The student clicks on the quiz link and selects "Start Quiz".	4. The system displays the quiz instructions, including time limits, grading policies. A timer starts when the quiz is launched.
5. The student answers the quiz questions (multiple-choice, short answer, etc.) by selecting or typing their responses.	6. The system saves the student’s progress periodically and tracks the time remaining.
7.The student reviews their answers and clicks "Submit Quiz" once all questions are answered (or the student decides to finish).	8.The system validates the submission (checks for unanswered required questions, ensures the quiz is submitted before the timer expires).
9.The student confirms the submission.	10.The system records the submission, provides a confirmation message, and automatically grades the objective based questions.

- **Postconditions:**
- The student submits the quiz successfully, and their answers are saved in the system.
- The quiz is automatically graded (for objective questions)
- The student receives a confirmation of submission.

Alternate Flow:

□ **Unanswered Questions**:

- If the student attempts to submit the quiz with unanswered required questions:
    - **System Response**: The system alerts the student: "You have unanswered questions. Are you sure you want to submit?" The student can choose to go back and complete the quiz or submit as-is.

□ **Timeout**:

- If the student’s time expires before they manually submit the quiz:
  - **System Response**: The system automatically submits the quiz on behalf of the student and displays a message: "Time is up. Your quiz has been submitted automatically."

□ **Technical Issue/Connection Loss**:

- If the student loses connection during the quiz or experiences a technical issue:
  - **System Response**: The system automatically saves progress at regular intervals (auto-save) and allows the student to resume from where they left off, provided the quiz window is still open.

<table>
<tr><th colspan="3">View Grades</th></tr>
<tr><td><strong>Actor:</strong></td><td colspan="2">Student</td></tr>
<tr><td><strong>Description:</strong></td><td colspan="2">The student can view their grades for completed quizzes, assignments, and exams.</td></tr>
<tr><td><strong>Preconditions:</strong></td><td colspan="2">• The student is logged into the system.
• The student has completed at least one assessment (quiz, assignment, etc.).
• The instructor has published the grades for the quiz or assignment.</td></tr>
<tr><td colspan="3"><strong>Flow of Events:</strong></td></tr>
<tr><td colspan="2"><strong>User Action</strong></td><td><strong>System Response</strong></td></tr>
<tr><td colspan="2">1. The student navigates to the course page and selects "View Grades."
3. The student selects a specific quiz or assignment to view detailed grades and feedback.</td><td>2. The system retrieves the student's grade records from the database.
4. The system displays the grade, including the total points, points earned, and any feedback provided by the instructor</td></tr>
<tr><td colspan="3"><strong>Postconditions:</strong>
• The student successfully views their grades and any feedback provided by the instructor.</td></tr>
<tr><td colspan="3"><strong>Alternate Flow:</strong>
□ <strong>Grades Not Published Yet</strong>:
• If the student selects an assessment for which grades are not yet available, the system displays a message: "Grades for this quiz/assignment have not been published yet."
□ <strong>Assessment Not Completed</strong>:
• If the student tries to view grades for an assessment they have not completed, the system displays a message: "You have not completed this quiz/assignment."</td></tr>
</table>

Create Course	
Actor:	Instructor
Description:	Instructor creates a new course by defining the syllabus, adding materials, and setting deadlines.
Preconditions:	Instructor must be logged in and have the appropriate privileges.
Flow of Events:	
User Action	**System Response**
1. The instructor clicks "Create Course", fills out the required information (e.g., course title, syllabus, deadlines), and clicks "Save".	2. The system validates the information (e.g., ensures all required fields are filled). 3. The system creates the course in the database. 4. The course is now listed in the course catalog and available for student enrollment.
Postconditions: The course is created and available for students to enroll.	
Alternate Flow: • **Missing Information**: The system checks the required fields. If any are missing, it displays an error: "Please complete all required fields before saving."	

Manage Course Materials	
Actor:	Instructor
Description:	Instructor can upload, update, and organize course content such as videos, readings, and assignments.
Preconditions:	Course must be created, and the instructor must be authorized.
Flow of Events:	
User Action	**System Response**
1. The instructor navigates to the course management page, uploads new content or modifies existing content, and clicks "Save".	2. The system validates the file types and sizes (if new content is uploaded). 3. The system updates the course materials in the database. 4. The system confirms the successful update and makes the new/updated materials available to students.
Postconditions: The updated course materials are accessible to enrolled students.	
Alternate Flow: • **Invalid File Format/Size**: • The system displays an error if the content format or size is invalid: "Unsupported file format. Please upload a valid file."	

<table>
<tr><th colspan="2">Grade Assignments/Quizes</th></tr>
<tr><td><b>Actor:</b></td><td>Instructor</td></tr>
<tr><td><b>Description:</b></td><td>Instructor reviews and grades student assignments.</td></tr>
<tr><td><b>Preconditions:</b></td><td>Students must have submitted their assignments.</td></tr>
<tr><td colspan="2"><b>Flow of Events:</b></td></tr>
<tr><td><b>User Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1. The instructor views the list of submitted assignments, selects a student submission, assigns a grade, and clicks "Submit Grade".</td><td>2. The system updates the student's grade in the database.
3. The system sends a notification to the student with their grade and feedback.</td></tr>
<tr><td colspan="2"><b>Postconditions:</b> Grades are assigned, and feedback is provided to students.</td></tr>
<tr><td colspan="2"><b>Alternate Flow:</b>
- <b>Grade Entry Error</b>:
○ If an invalid grade is entered (e.g., out of the acceptable range), the system displays an error message: "Invalid grade. Please enter a valid score.</td></tr>
</table>

<table>
<tr><th colspan="2">Communicate with Students</th></tr>
<tr><td><b>Actor:</b></td><td>Instructor</td></tr>
<tr><td><b>Description:</b></td><td>Instructor communicates with students via announcements, emails, or discussion forums.</td></tr>
<tr><td><b>Preconditions:</b></td><td>Communication channels must be active.</td></tr>
<tr><td colspan="2"><b>Flow of Events:</b></td></tr>
<tr><td><b>User Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1. The instructor types an announcement or message and clicks "Send" or "Post".</td><td>2. The system broadcasts the message to the relevant students (via email or notification).
3. The message appears in the relevant course page and inbox for the students.</td></tr>
<tr><td colspan="2"><b>Postconditions:</b> Message is successfully delivered to students.</td></tr>
</table>

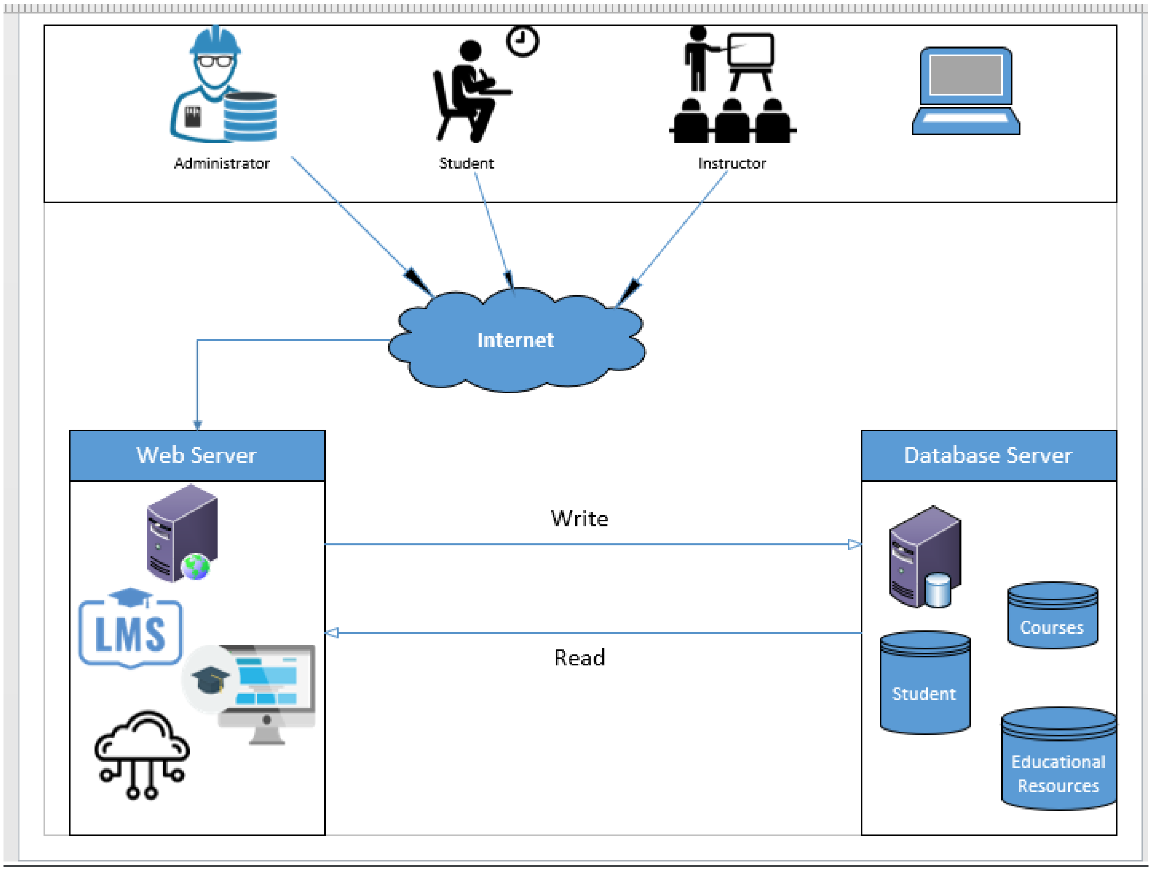
Create Quizes	
Actor:	Instructor
Description:	The instructor creates and publishes quizzes for students. This includes defining questions, setting time limits, assigning point values, and determining availability dates.
Preconditions:	• The instructor is logged into the system. • A course exists, and the instructor has the appropriate permissions to create a quiz. • The course must be active, and students are enrolled in it.
Flow of Events:	
User Action	**System Response**
1. The instructor navigates to the course management page and selects "Create Quiz." 3. The instructor enters quiz details (title, instructions, time limit, etc.). 5. The instructor adds questions to the quiz (multiple-choice, short answer, etc.). 7.The instructor sets the availability dates and any grading rules (e.g., automatic or manual grading). The instructor clicks "Publish."	2.The system displays the quiz creation interface. 4.The system validates the inputs for required fields. 6. The system verifies that the questions are correctly formatted and saves them. 8.The system confirms the dates and grading settings. 9.The quiz is made available to students during the specified time window, and notifications are sent to enrolled students.
Postconditions: • The quiz is successfully created and made available to enrolled students. • Students can access the quiz within the defined availability period.	
Alternate Flow: **Missing Required Fields**: • The system checks if any required information (e.g., quiz title or time limit) is missing.	

Manage Users	
Actor:	Admin
Description:	Admin adds, updates, or deletes user accounts (students, instructors).
Preconditions:	Admin must be logged in.
Flow of Events:	
User Action	**System Response**
1. The admin selects a user from the user list, modifies their information (e.g., role, access privileges), and clicks "Save".	2. The system updates the user's information in the database. 3. The system confirms the changes and logs the action.
Postconditions: User accounts are updated in the system.	
Alternate Flow: • **Invalid Input**: o If invalid data is entered (e.g., incorrect email format), the system displays an error: "Invalid input. Please correct the highlighted fields."	

Manage Platform Settings	
Actor:	Admin
Description:	Admin configures platform-wide settings such as course policies, system maintenance, and backups.
Preconditions:	Admin must have appropriate privileges.
Flow of Events:	
User Action	**System Response**
1. The admin navigates to the platform settings page, modifies system-wide settings (e.g., user registration policies, backup schedules), and clicks "Save".	2. The system updates the settings in the database. 3. The changes take effect immediately or at a scheduled time, depending on the type of change.
Postconditions: System settings are updated and applied.	

Monitor System	
Actor:	Admin
Description:	Admin monitors platform performance, usage, and security.
Preconditions:	Admin dashboard should be functional
Flow of Events:	
User Action	**System Response**
1. The admin accesses the system monitoring dashboard to view metrics (e.g., server performance, user activity, security logs).	2. The system retrieves and displays the relevant data in real-time. 3. The admin can take necessary actions based on the metrics (e.g., adjusting server capacity, reviewing security alerts).
Postconditions: Admin has insights into system operations.	

Architecture Diagram

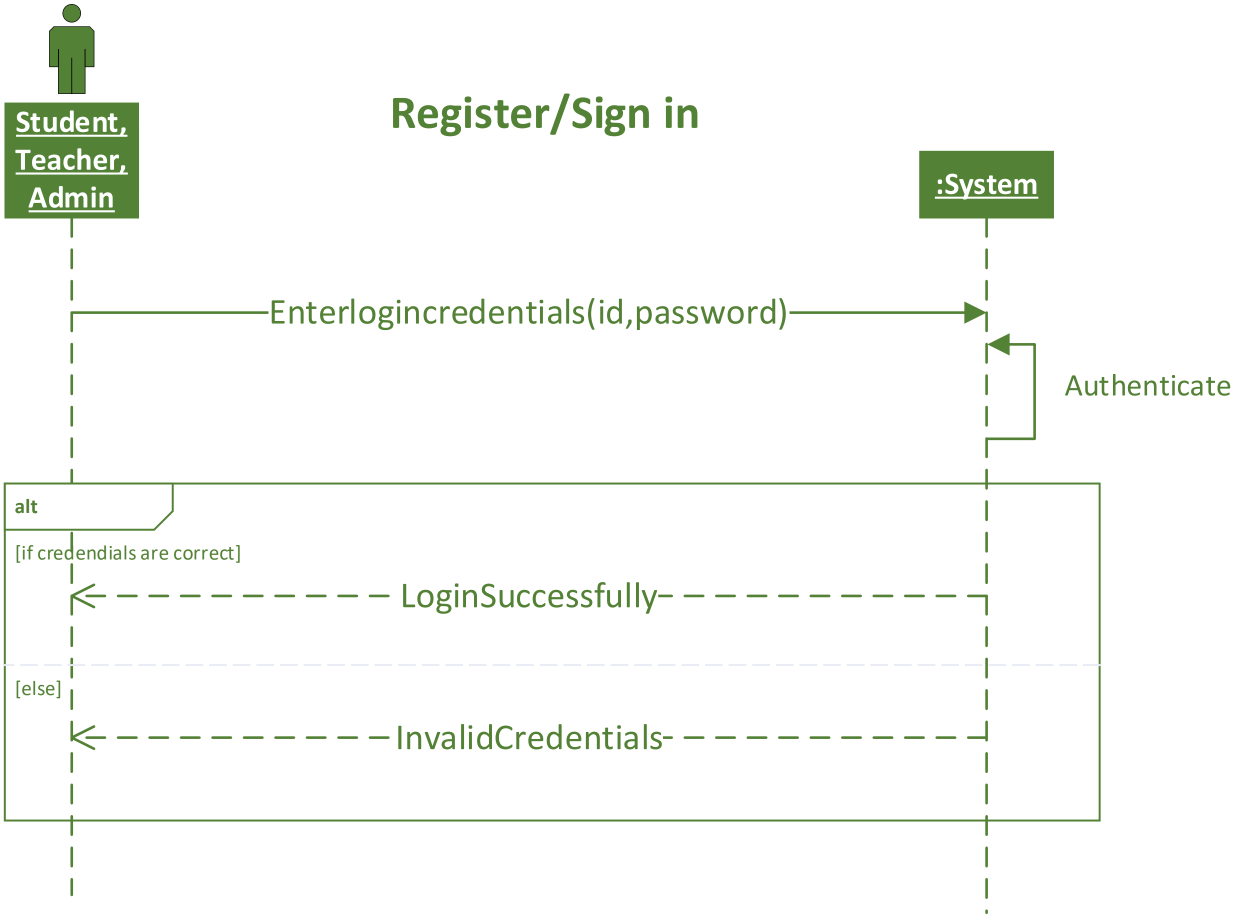


Three tier architecture of e-learning platform

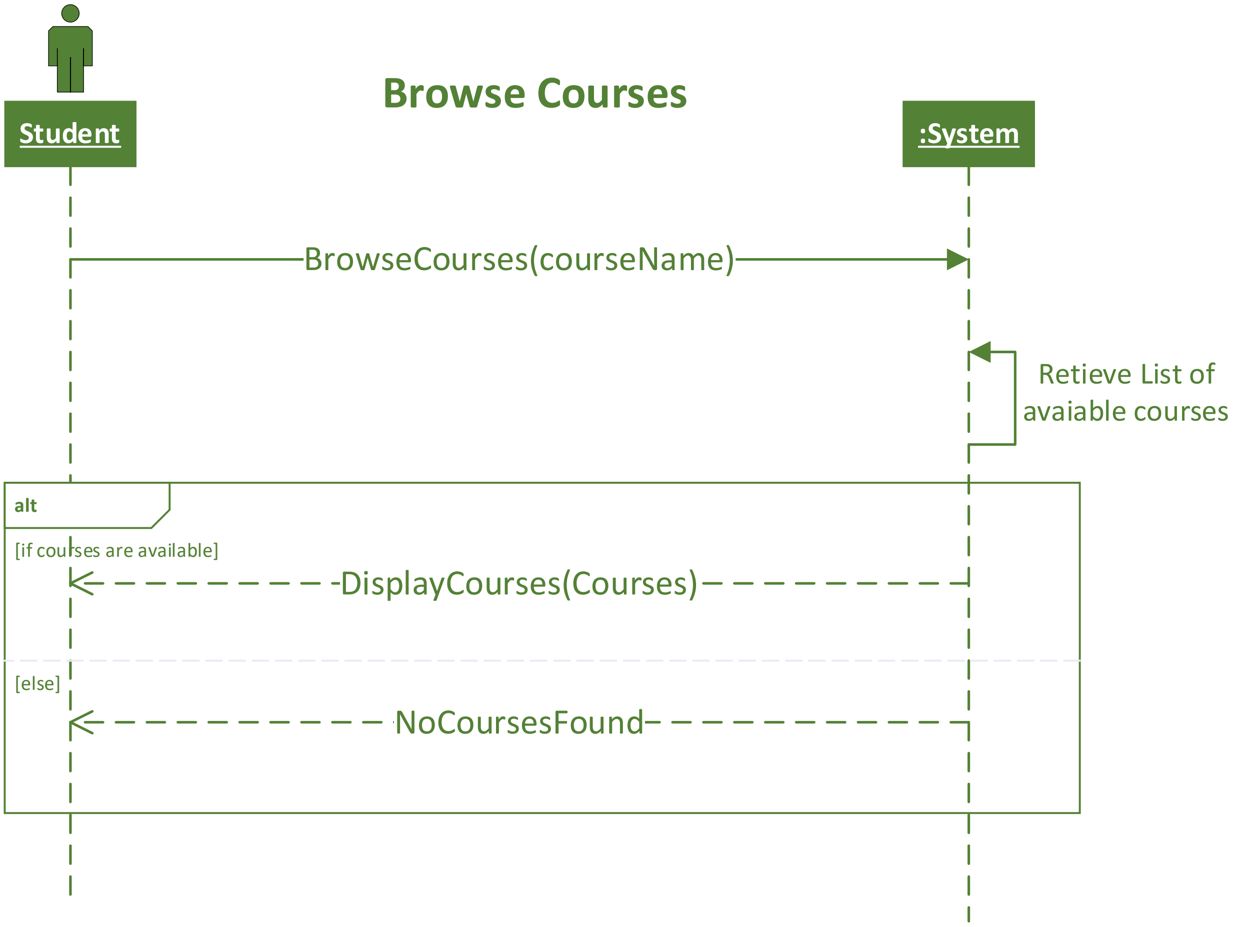
System Sequence Diagrams

For Student Entity:

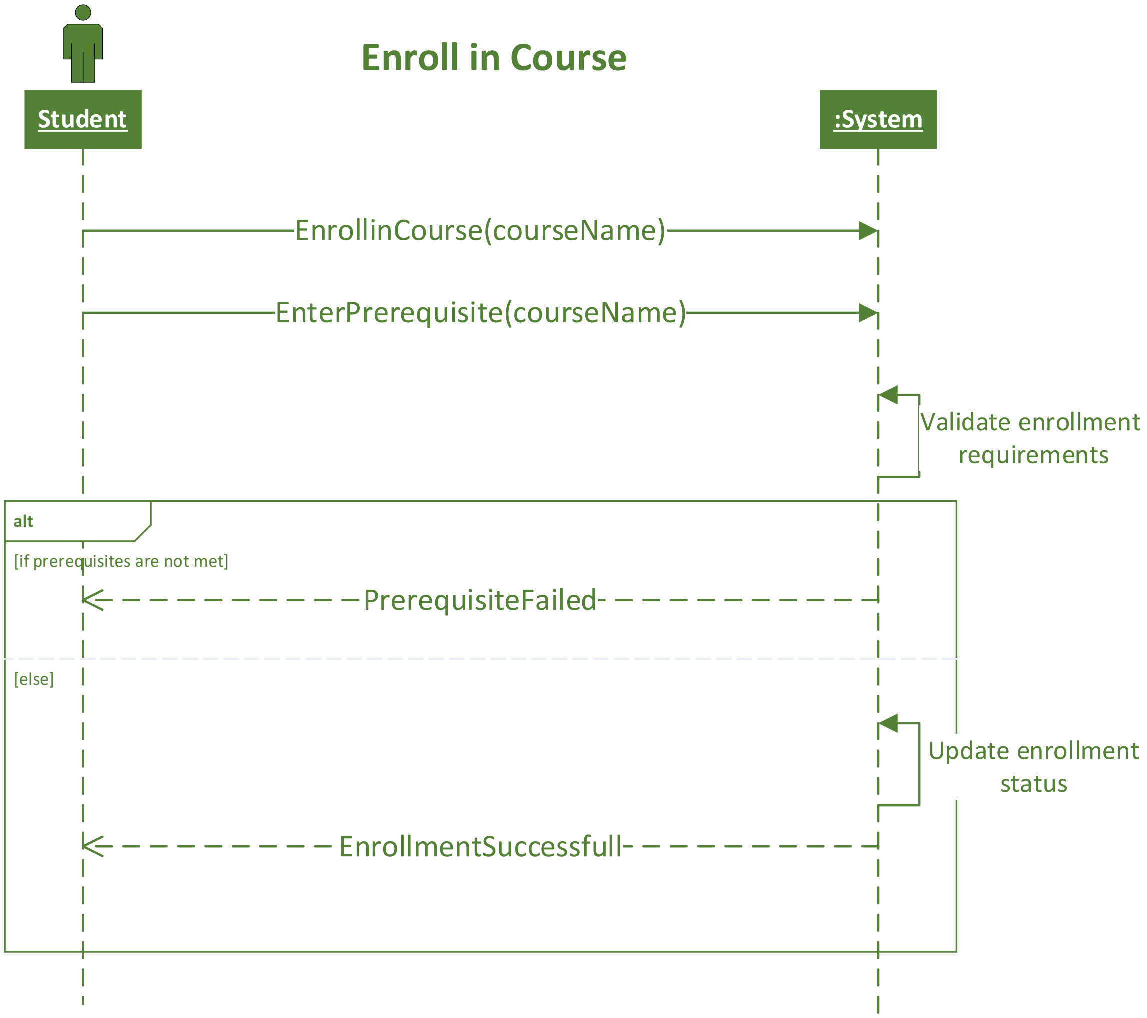
- **Sign in:**



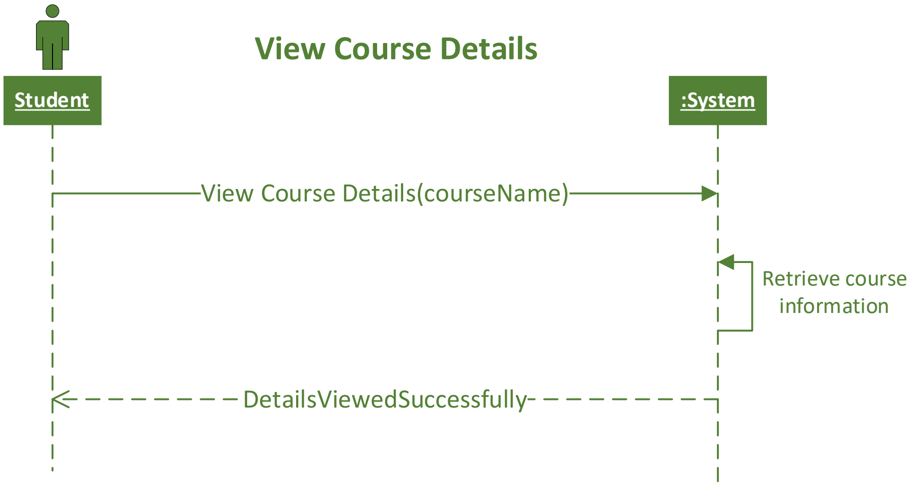
- **Browse Course:**


Browse Courses
Student
:System
BrowseCourses(courseName)
Retieve List of avaiable courses
alt
[if courses are available]
DisplayCourses(Courses)
[else]
NoCoursesFound

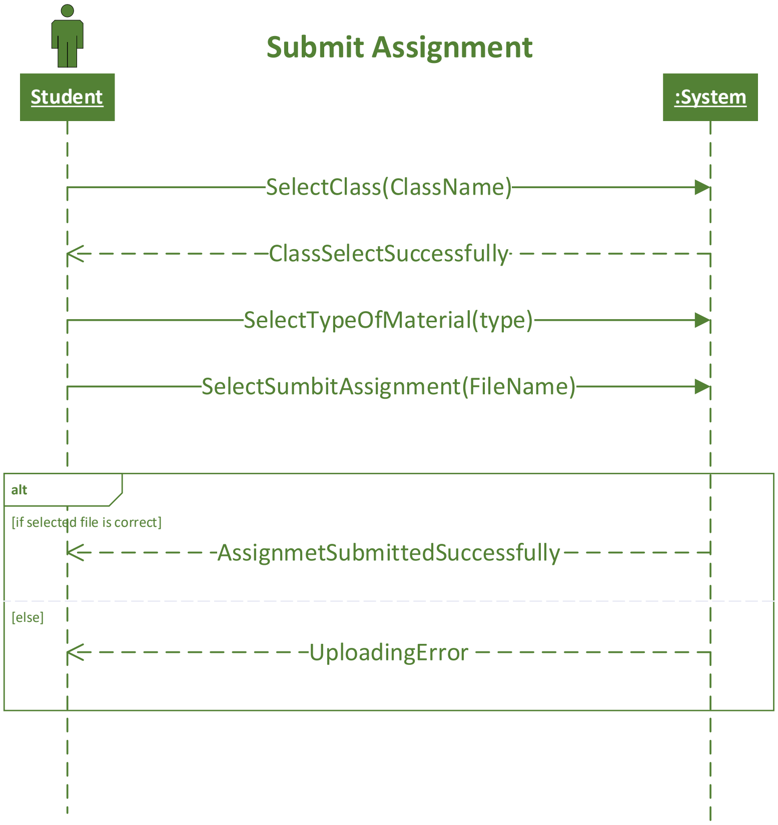
- **Enroll in Course:**


Enroll in Course
Student
:System
EnrollinCourse(courseName)
EnterPrerequisite(courseName)
Validate enrollment requirements
alt
[if prerequisites are not met]
PrerequisiteFailed
[else]
Update enrollment status
EnrollmentSuccessfull

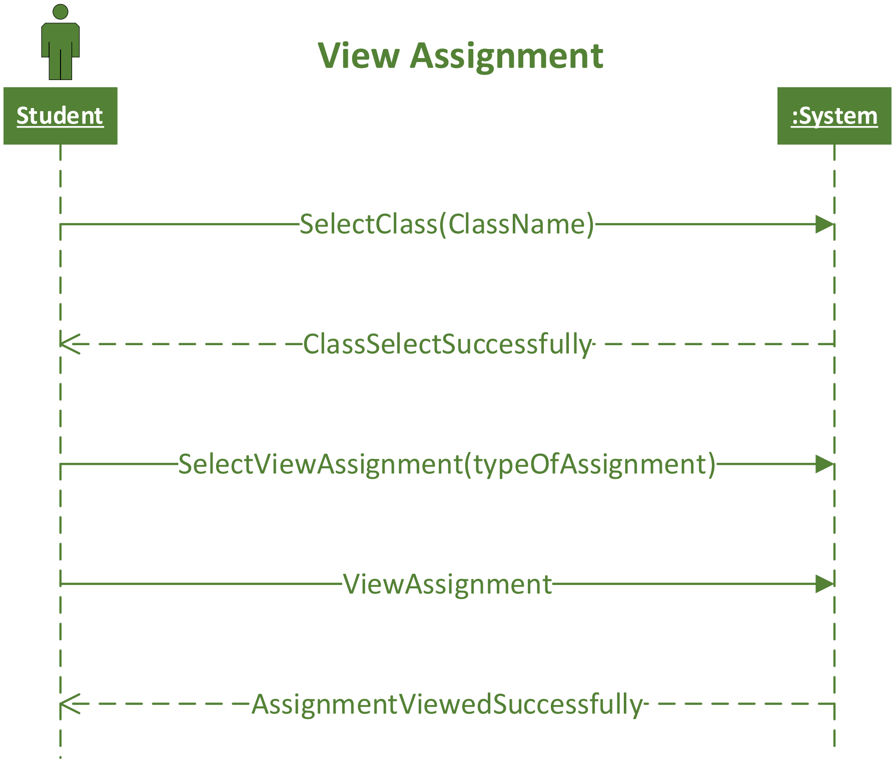
- **View Course Details:**


View Course Details
Student
:System
View Course Details(courseName)
Retrieve course information
DetailsViewedSuccessfully

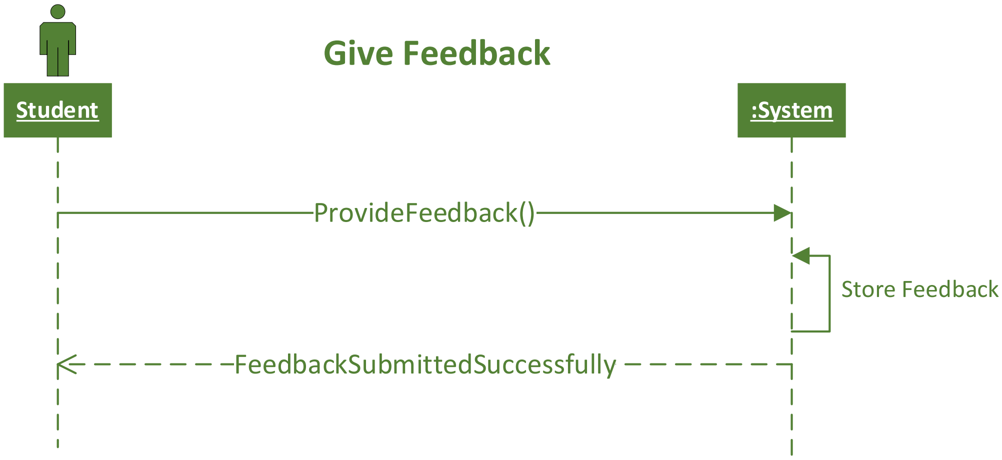
- **Submit Assignment:**


Submit Assignment
Student
:System
SelectClass(ClassName)
ClassSelectSuccessfully
SelectTypeOfMaterial(type)
SelectSumbitAssignment(FileName)
alt
[if selected file is correct]
AssignmetSubmittedSuccessfully
[else]
UploadingError

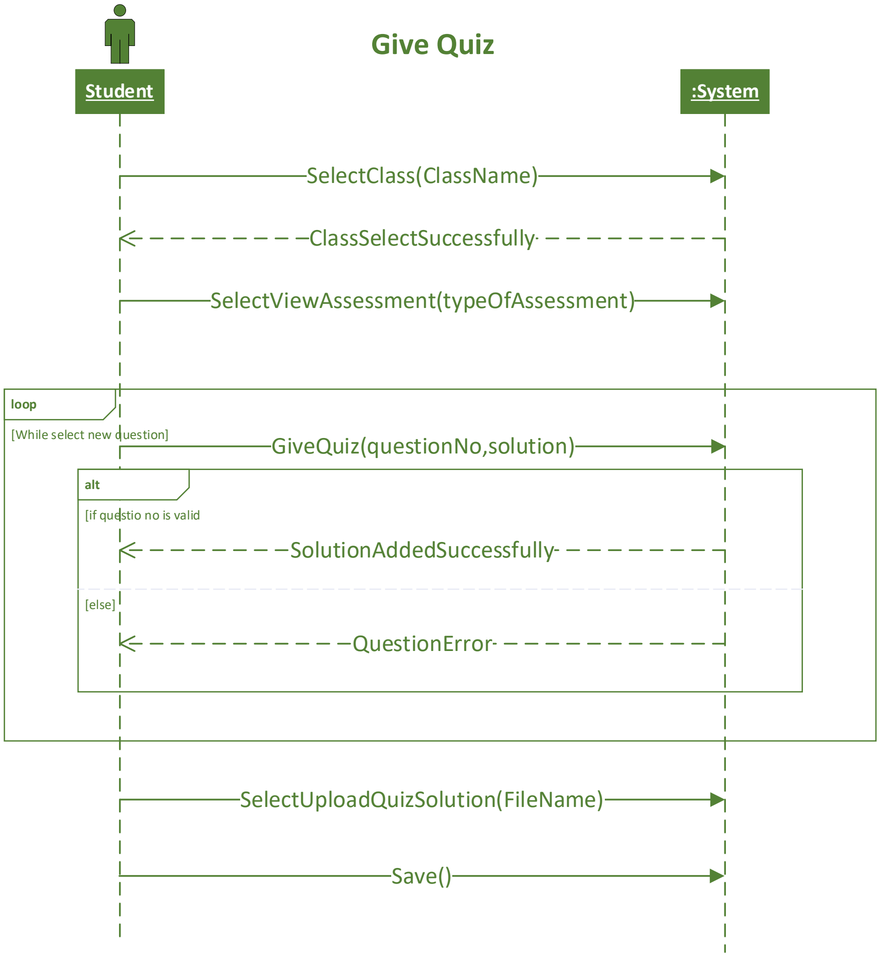
- **View Assignment:**



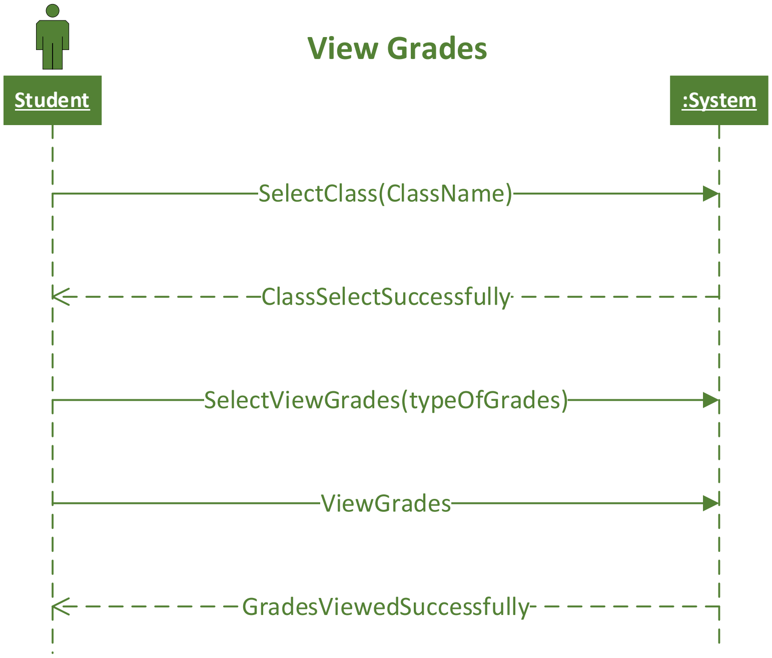
- **Give Feedback:**



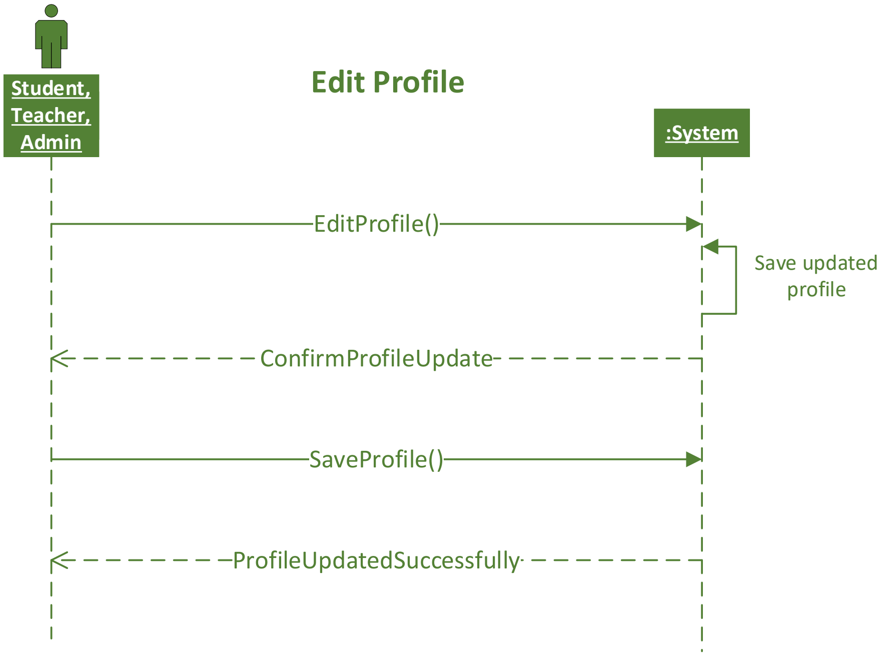
- **Give Quiz:**


Give Quiz
Student
:System
SelectClass(ClassName)
ClassSelectSuccessfully
SelectViewAssessment(typeOfAssessment)
loop
[While select new question]
GiveQuiz(questionNo,solution)
alt
[if questio no is valid
SolutionAddedSuccessfully
[else]
QuestionError
SelectUploadQuizSolution(FileName)
Save()

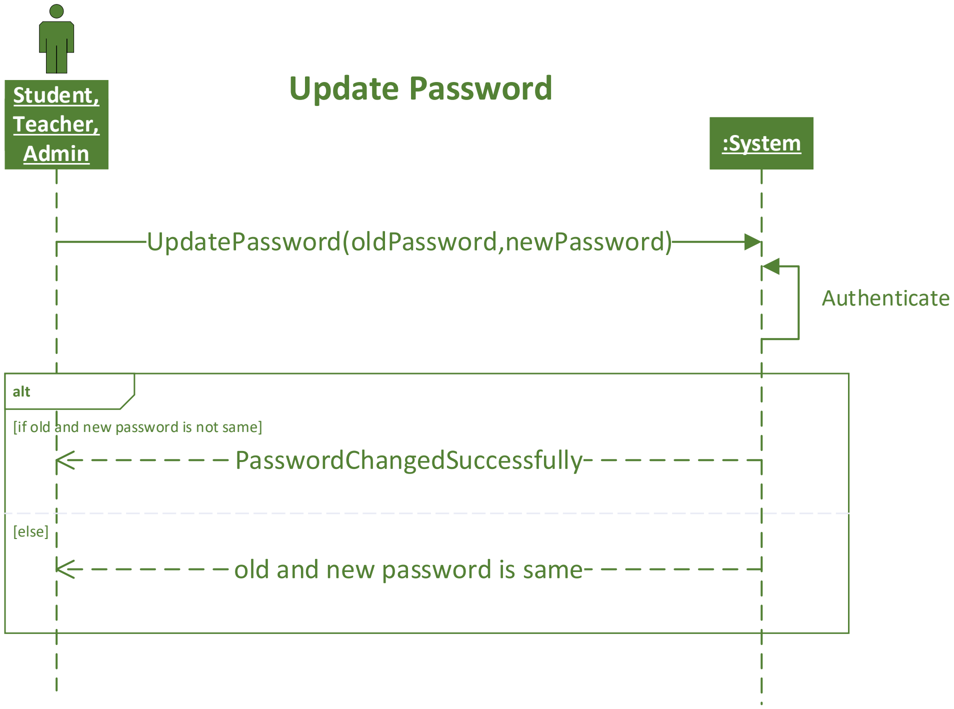
- **View Grades:**



- **Edit Profile:**

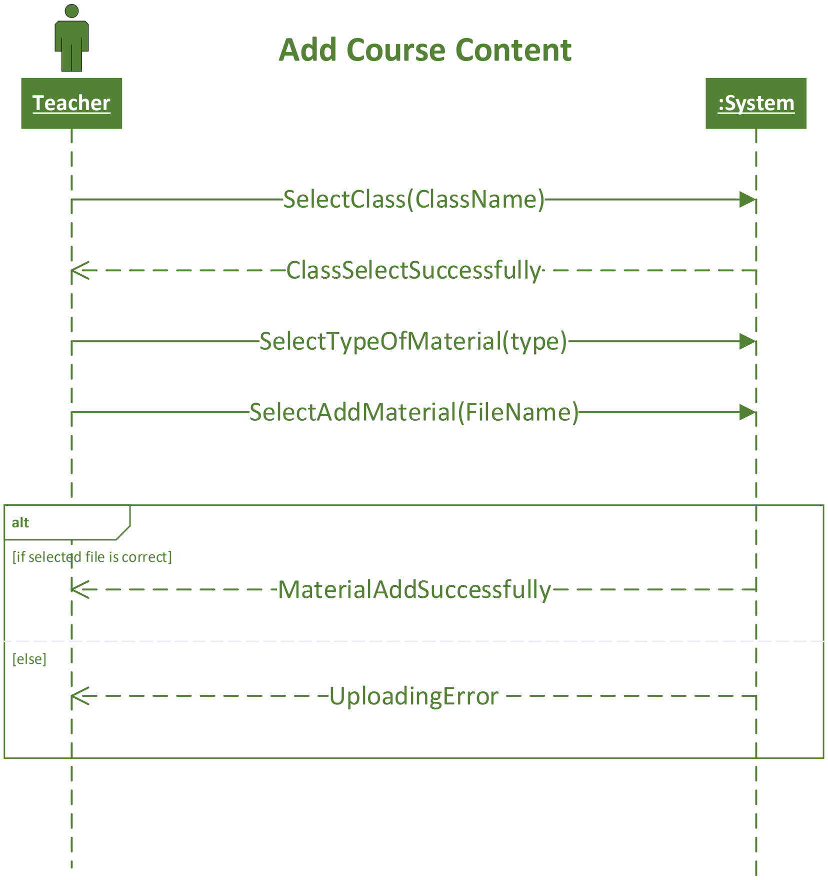


- **Change Password:**

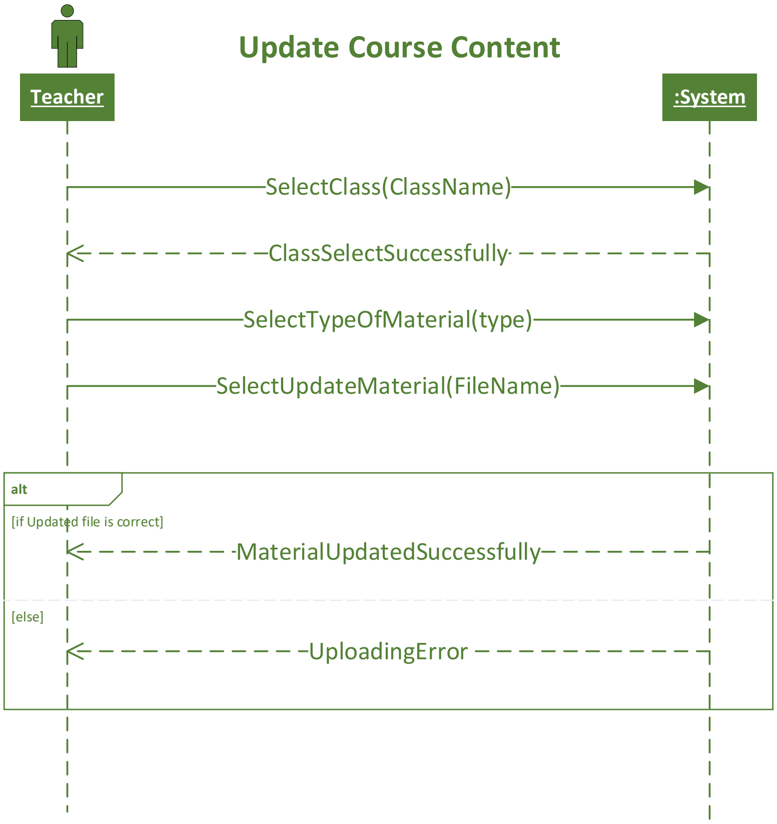

Student, Teacher, Admin
Update Password
:System
UpdatePassword(oldPassword,newPassword)
Authenticate
alt
[if old and new password is not same]
PasswordChangedSuccessfully
[else]
old and new password is same

For Teacher Entity:

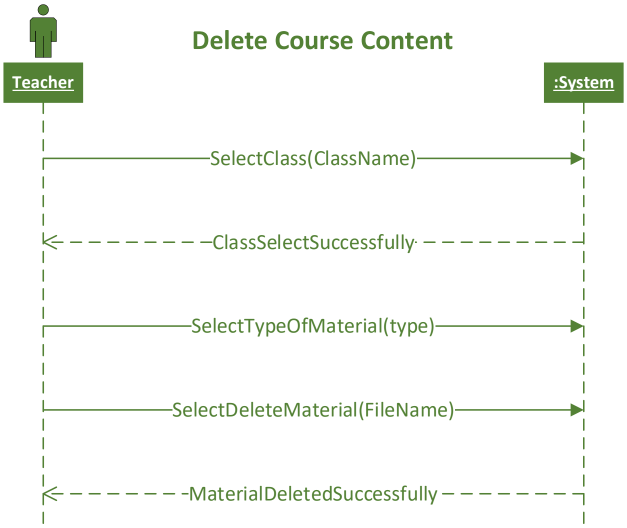
- **Add Course Content:**



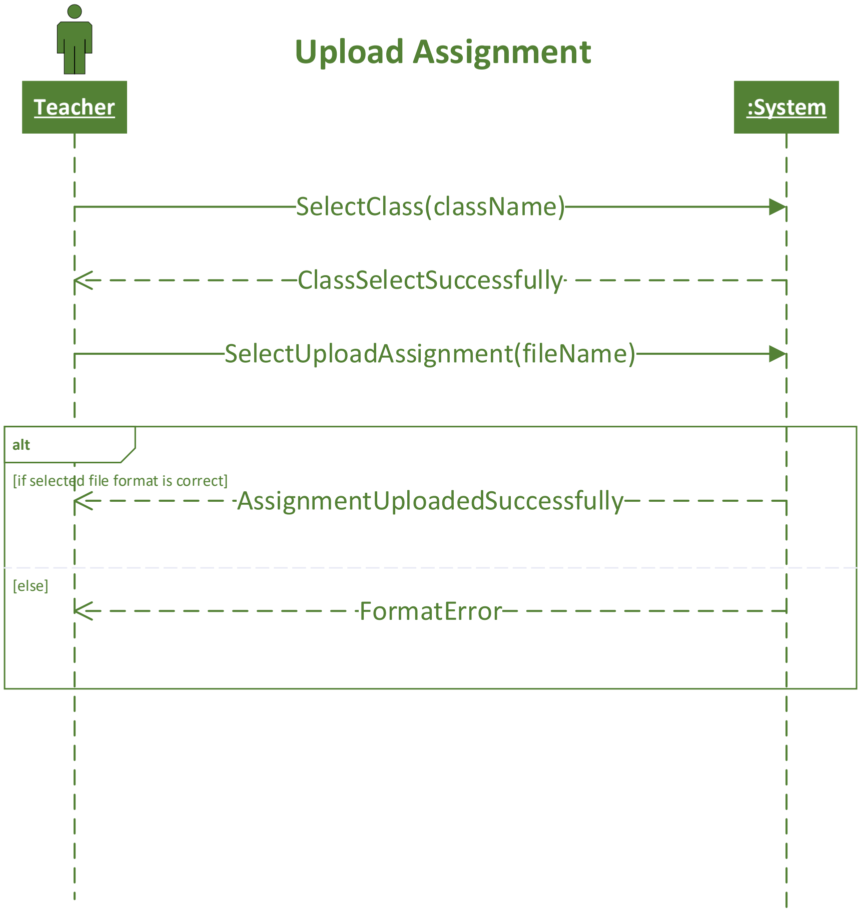
- **Update Course Content:**


Update Course Content
Teacher
:System
SelectClass(ClassName)
ClassSelectSuccessfully
SelectTypeOfMaterial(type)
SelectUpdateMaterial(FileName)
alt
[if Updated file is correct]
MaterialUpdatedSuccessfully
[else]
UploadingError

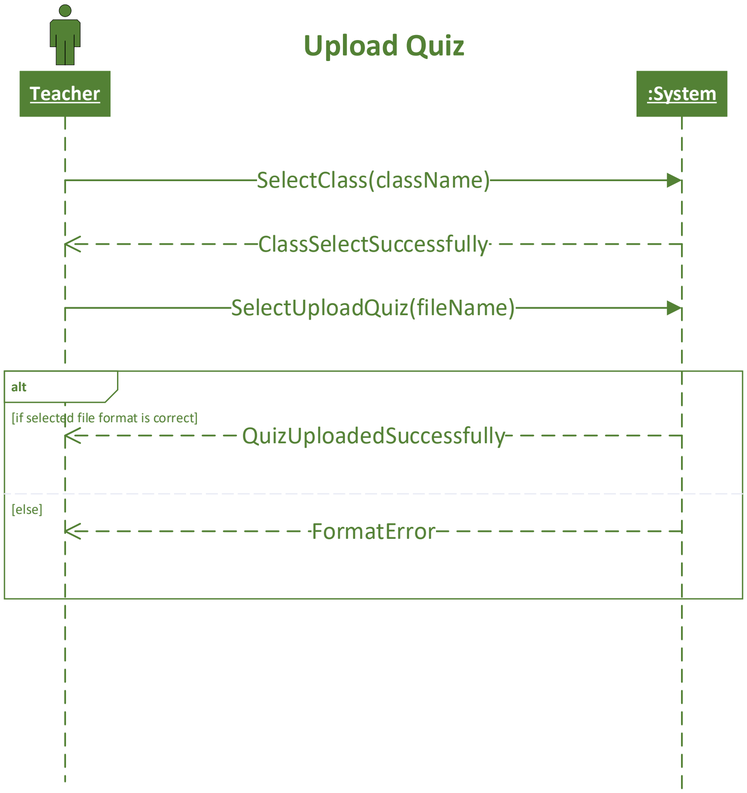
- **Delete Course Content:**


Delete Course Content
Teacher
:System
SelectClass(ClassName)
ClassSelectSuccessfully
SelectTypeOfMaterial(type)
SelectDeleteMaterial(FileName)
MaterialDeletedSuccessfully

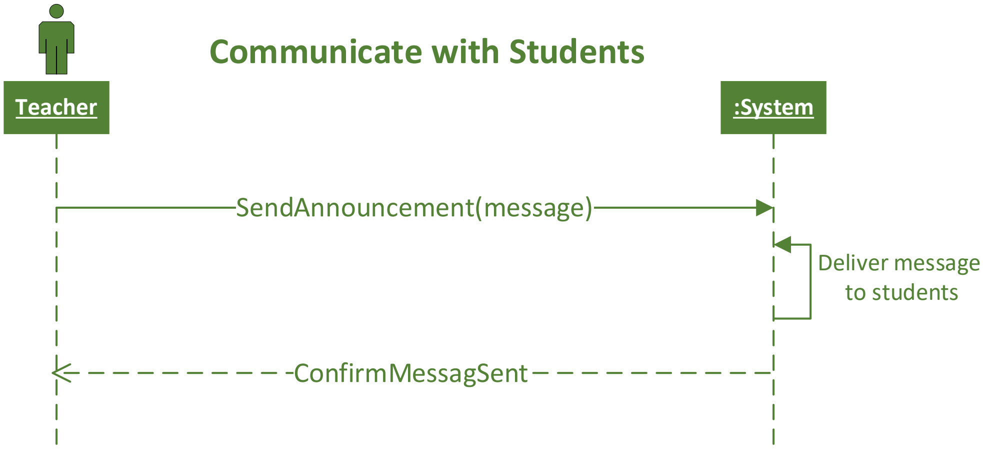
- **Upload Assignment:**


Upload Assignment
Teacher
:System
SelectClass(className)
ClassSelectSuccessfully
SelectUploadAssignment(fileName)
alt
[if selected file format is correct]
AssignmentUploadedSuccessfully
[else]
FormatError

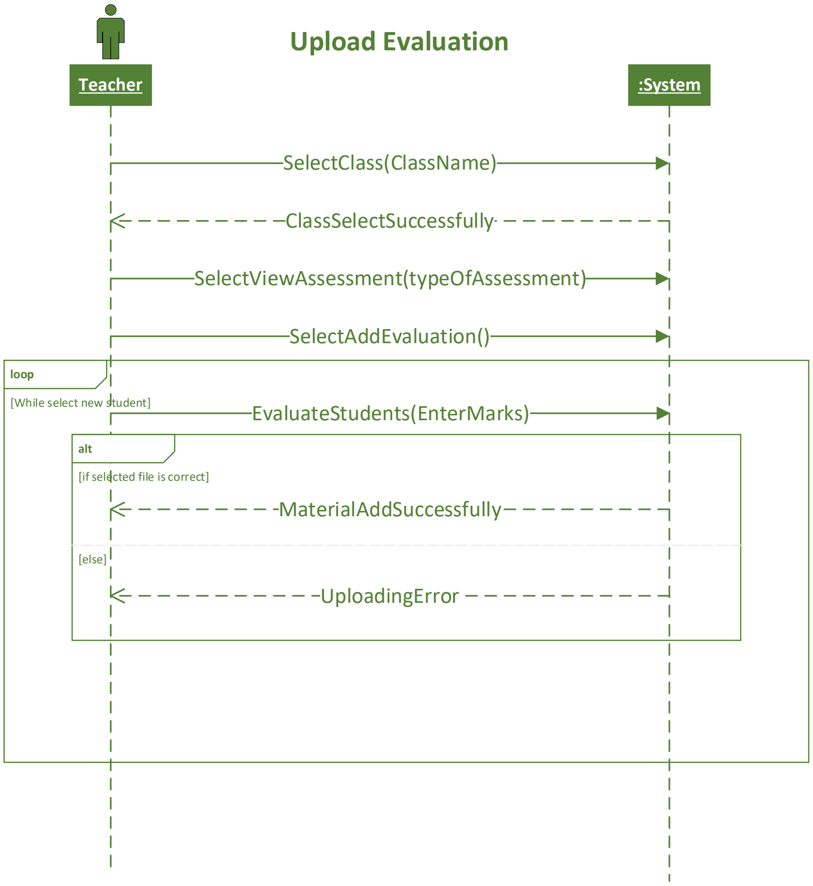
- **Upload Quiz:**



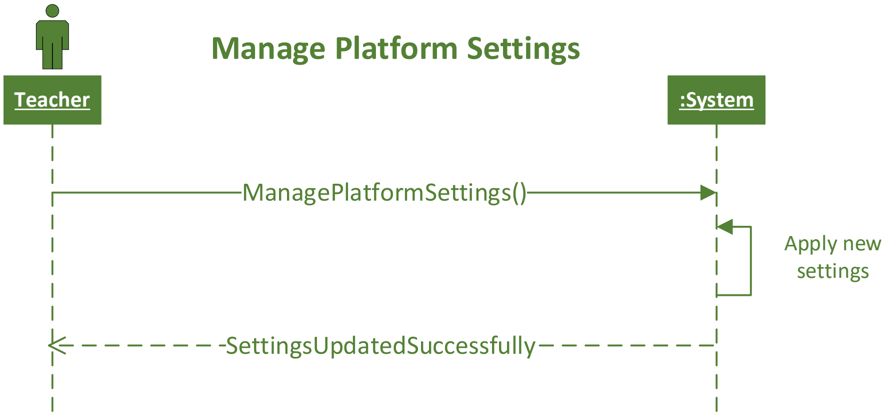
- **Communicate With Students:**



- **Upload Evaluation:**

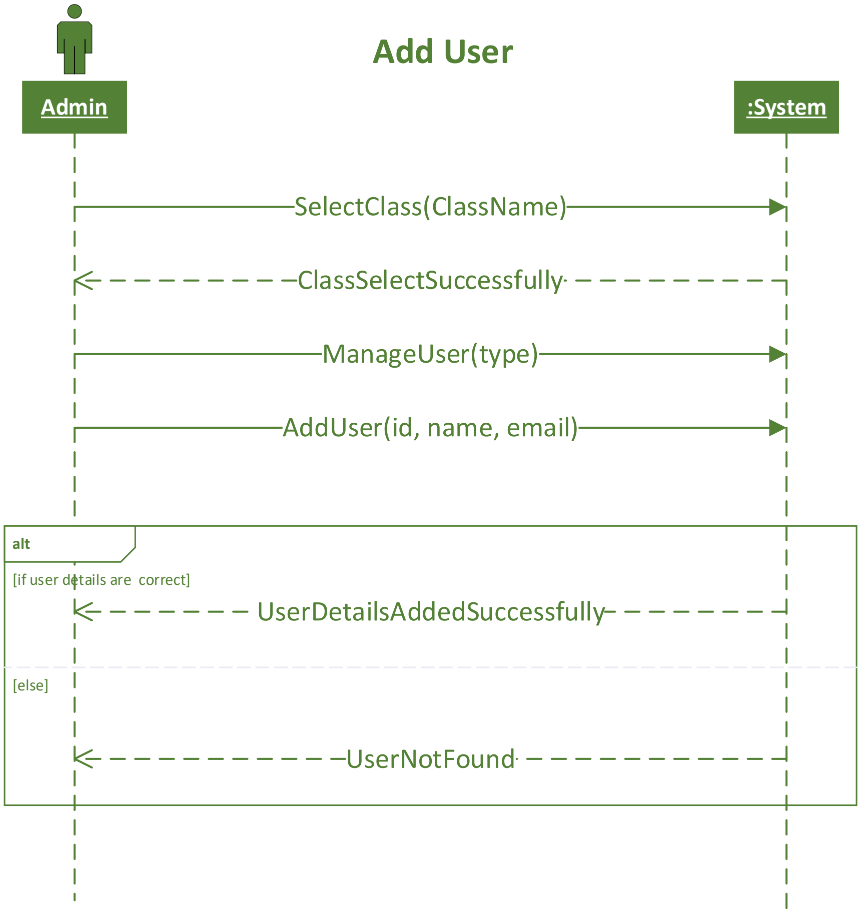


- **Manage Platform Settings:**

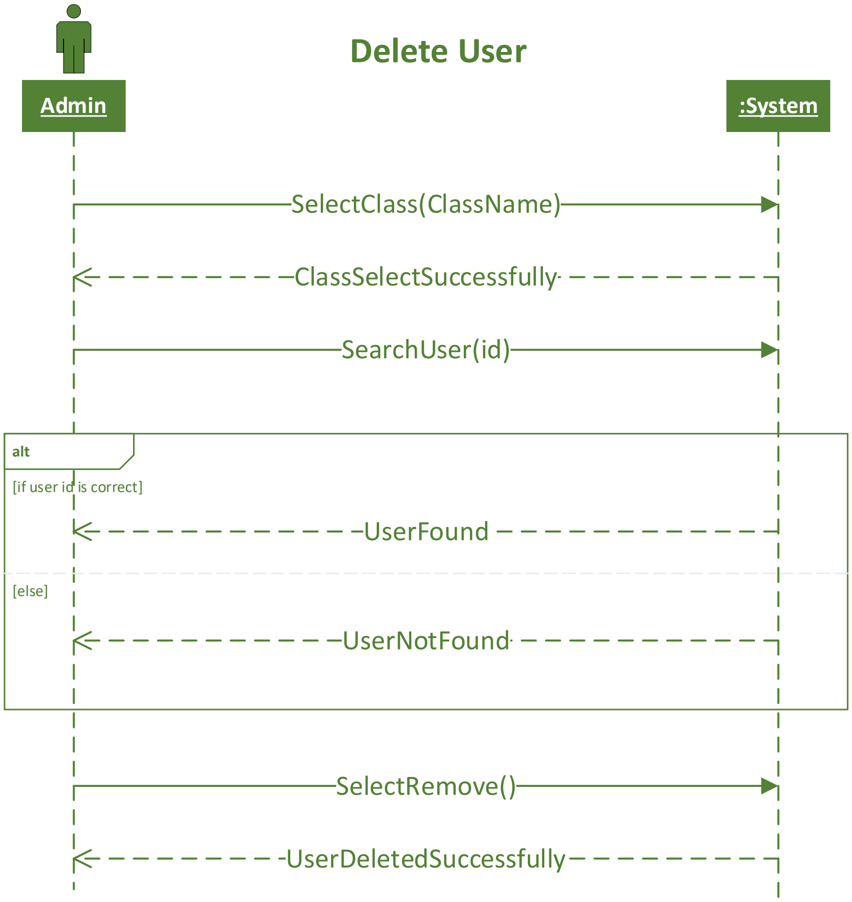


For Admin Entity:

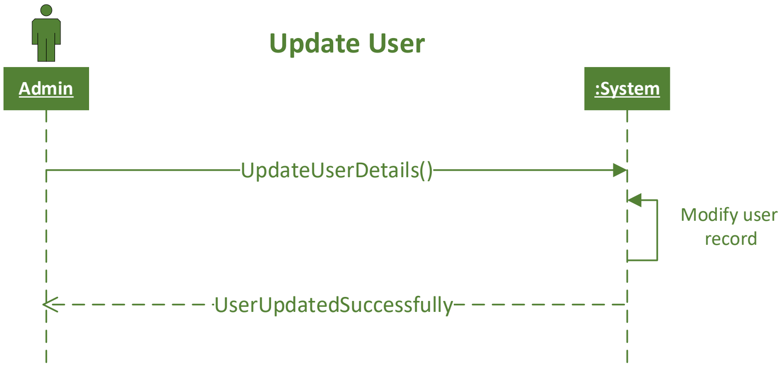
- **Add User:**



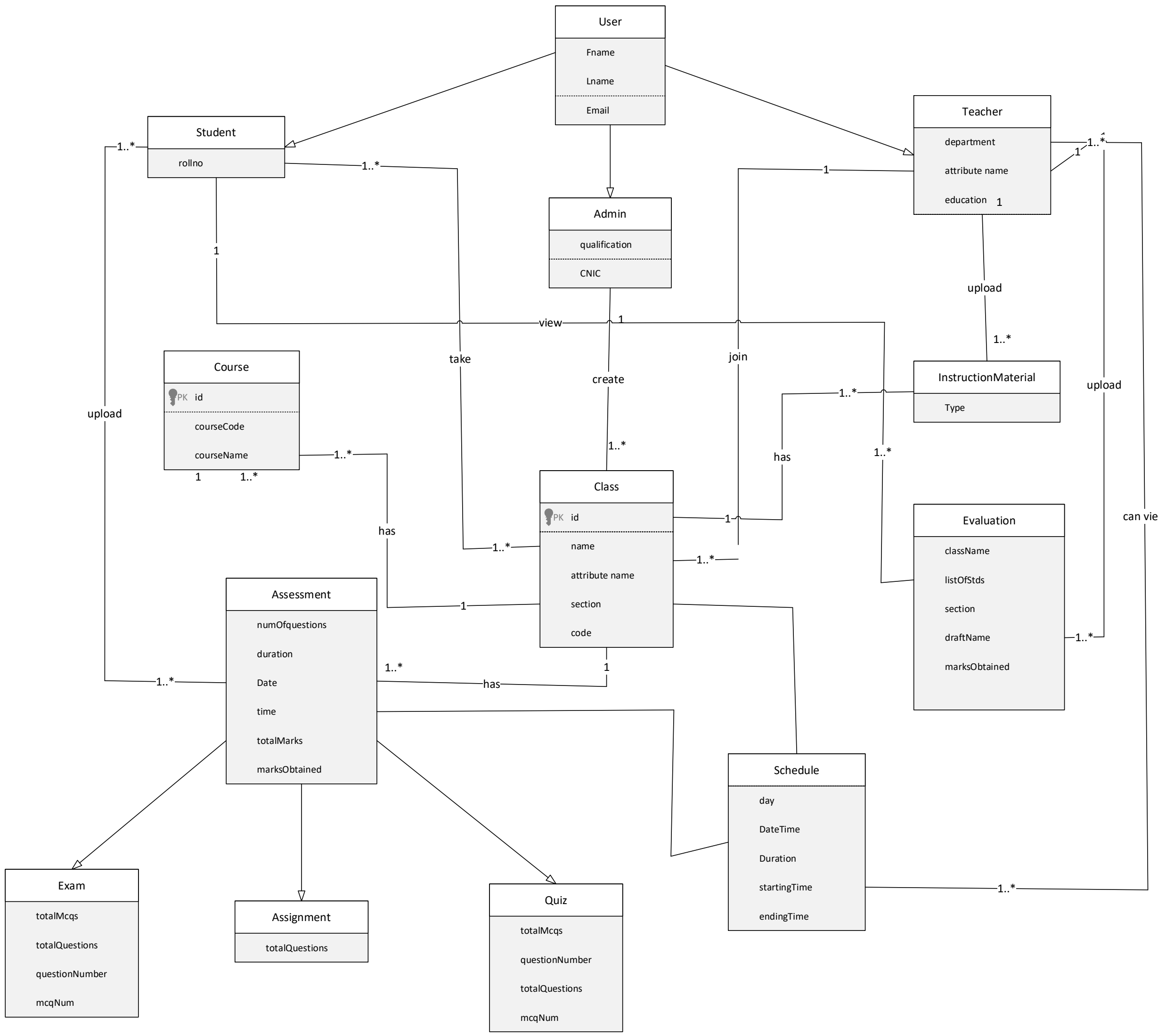
Delete User:



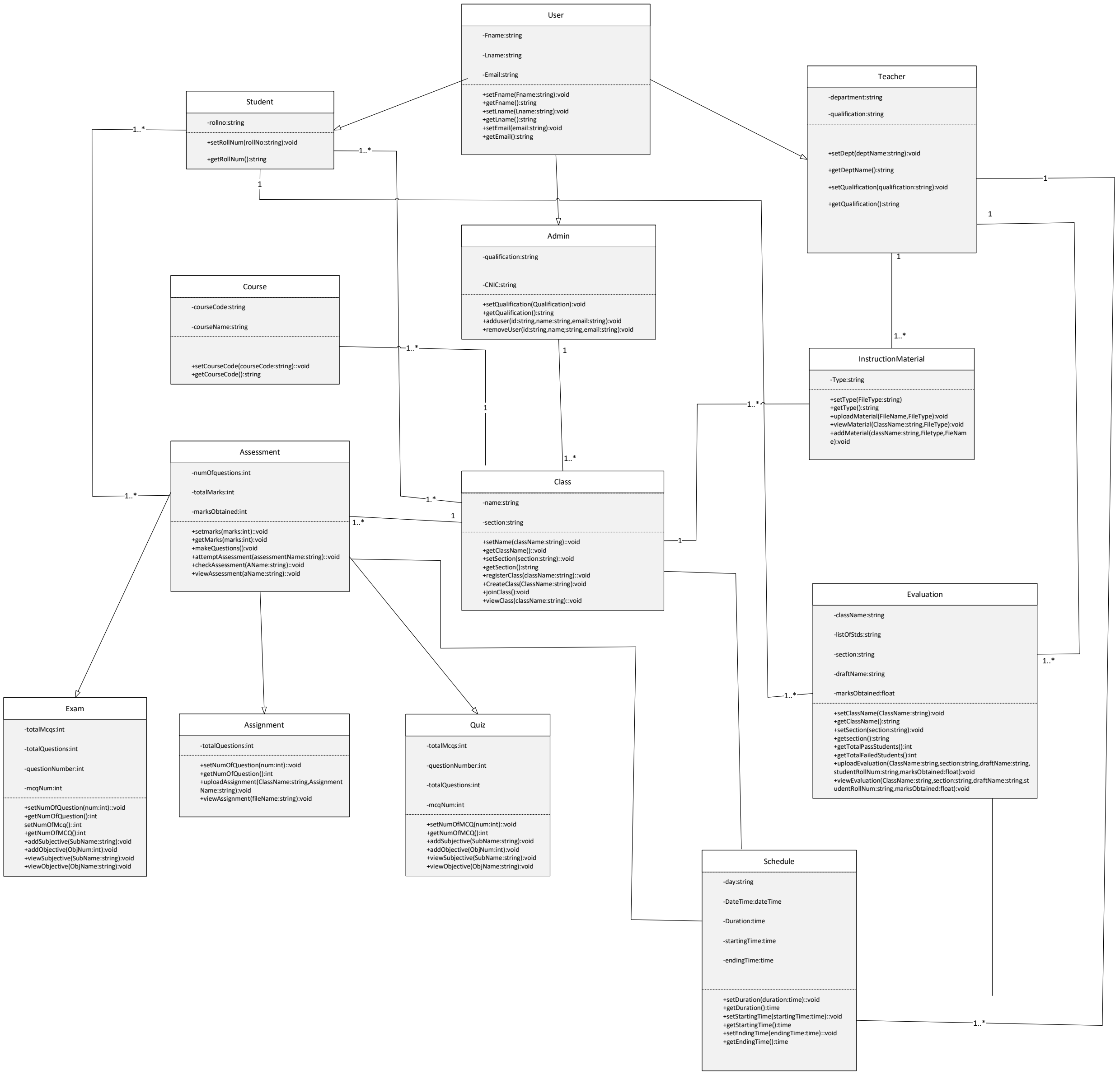
- **Update User:**



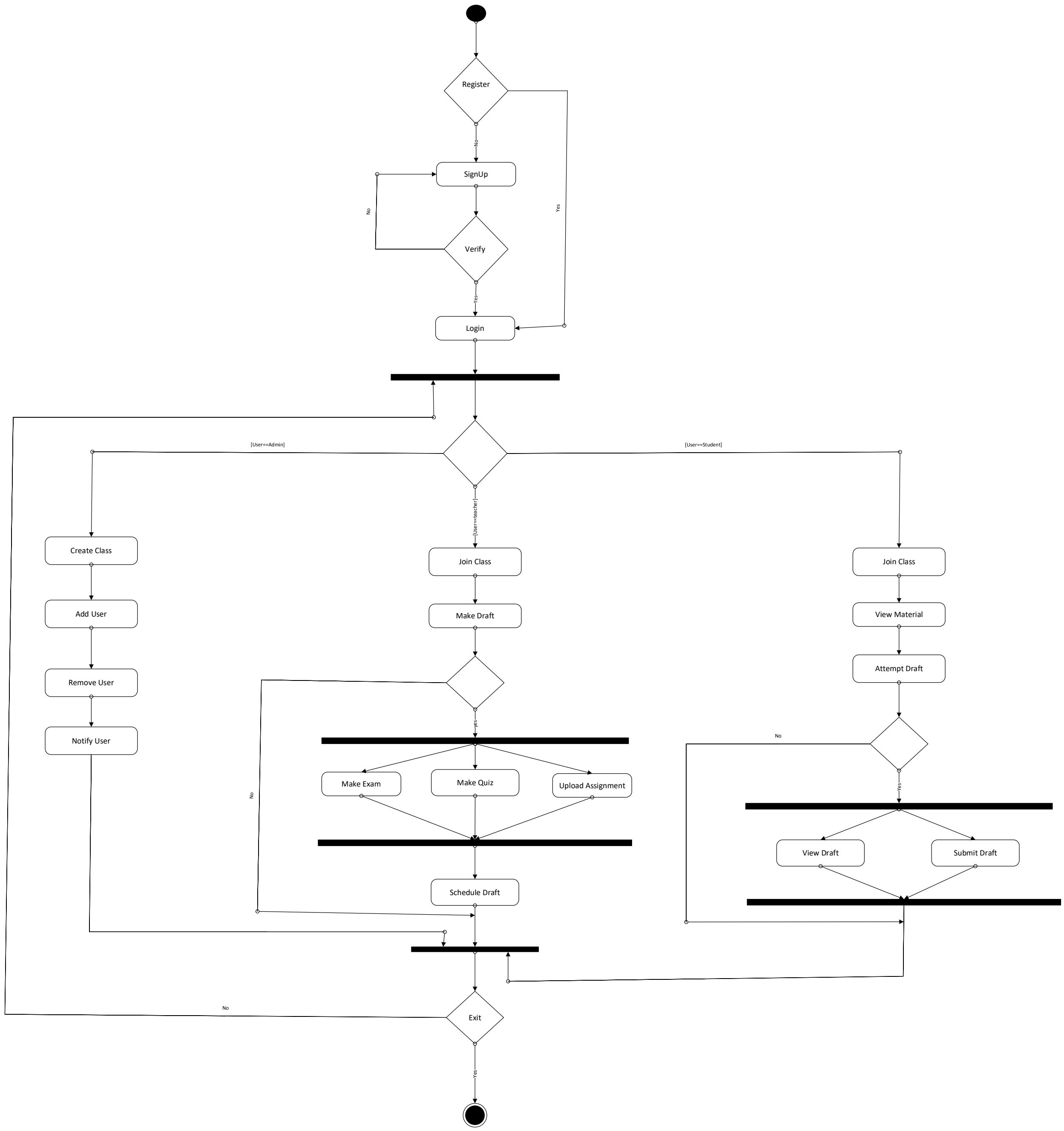
Domain Model



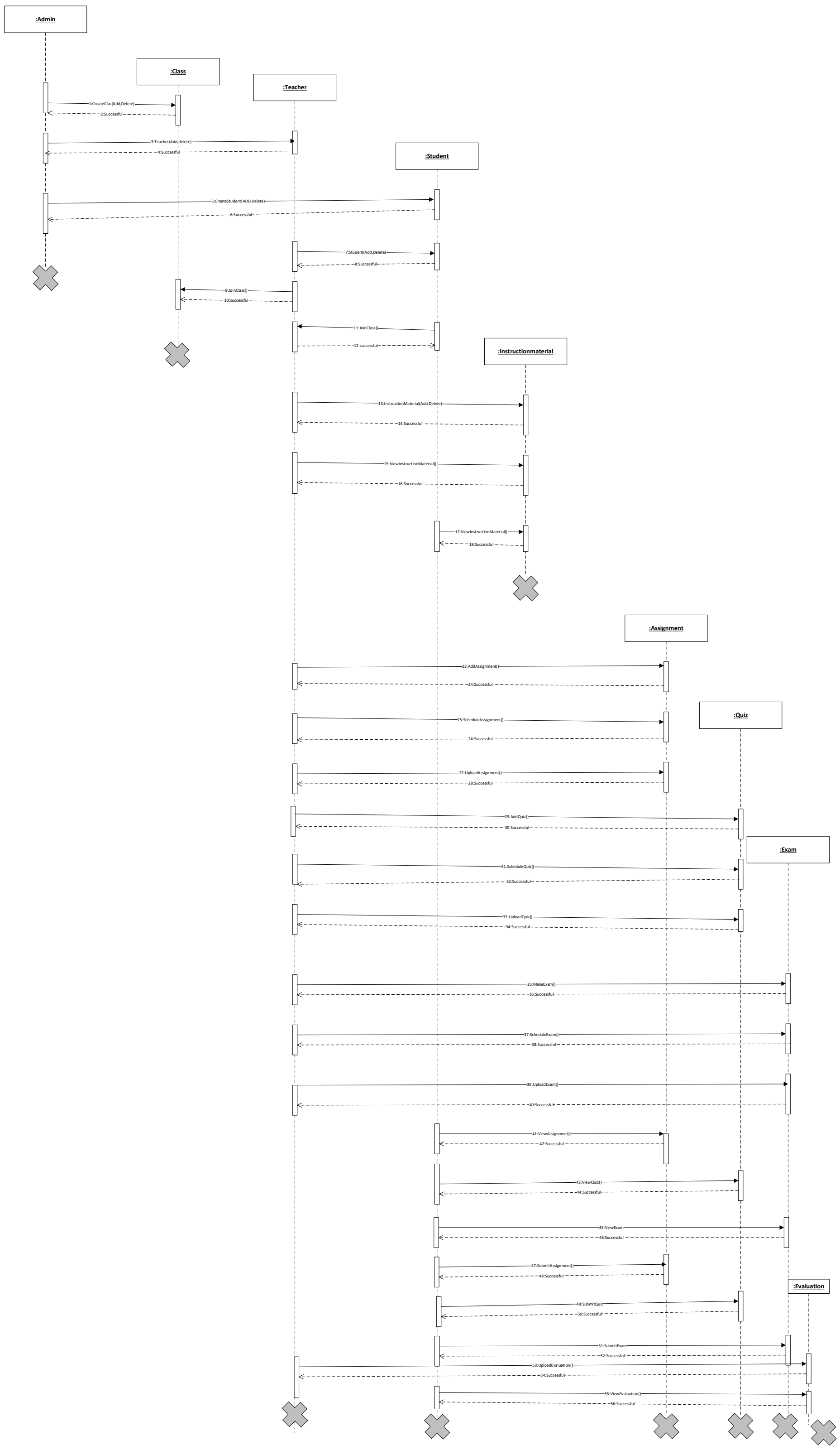
Class Diagram


User
-Fname:string
-Lname:string
-Email:string
+setFname(Fname:string):void
+getFname():string
+setLname(Lname:string):void
+getLname():string
+setEmail(email:string):void
+getEmail():string
Student
-rollno:string
+setRollNum(rollNo:string):void
+getRollNum():string
Teacher
-department:string
-qualification:string
+setDept(deptName:string):void
+getDeptName():string
+setQualification(qualification:string):void
+getQualification():string
Admin
-qualification:string
-CNIC:string
+setQualification(Qualification):void
+getQualification():string
+adduser(id:string,name:string,email:string):void
+removeUser(id:string,name;string,email:string):void
Course
-courseCode:string
-courseName:string
+setCourseCode(courseCode:string)::void
+getCourseCode():string
InstructionMaterial
-Type:string
+setType(FileType:string)
+getType():string
+uploadMaterial(FileName,FileType):void
+viewMaterial(ClassName:string,FileType):void
+addMaterial(className:string,Filetype,FieName e):void
Assessment
-numOfquestions:int
-totalMarks:int
-marksObtained:int
+setmarks(marks:int)::void
+getMarks(marks:int):void
+makeQuestions():void
+attemptAssessment(assessmentName:string)::void
+checkAssessment(AName:string)::void
+viewAssessment(aName:string)::void
Class
-name:string
-section:string
+setName(className:string)::void
+getClassName()::void
+setSection(section:string)::void
+getSection():string
+registerClass(className:string)::void
+CreateClass(ClassName:string):void
+joinClass():void
+viewClass(className:string)::void
Evaluation
-className:string
-listOfStds:string
-section:string
-draftName:string
-marksObtained:float
+setClassName(ClassName:string):void
+getClassName():string
+setSection(section:string):void
+getsection():string
+getTotalPassStudents():int
+getTotalFailedStudents():int
+uploadEvaluation(ClassName:string,section:string,draftName:string, studentRollNum:string,marksObtained:float):void
+viewEvaluation(ClassName:string,section:string,draftName:string,st udentRollNum:string,marksObtained:float):void
Exam
-totalMcqs:int
-totalQuestions:int
-questionNumber:int
-mcqNum:int
+setNumOfQuestion(num:int)::void
+getNumOfQuestion():int
setNumOfMcq()::int
+getNumOfMCQ():int
+addSubjective(SubName:string):void
+addObjective(ObjNum:int):void
+viewSubjective(SubName:string):void
+viewObjective(ObjName:string):void
Assignment
-totalQuestions:int
+setNumOfQuestion(num:int)::void
+getNumOfQuestion():int
+uploadAssignment(ClassName:string,Assignment Name:string):void
+viewAssignment(fileName:string):void
Quiz
-totalMcqs:int
-questionNumber:int
-totalQuestions:int
-mcqNum:int
+setNumOfMCQ(num:int)::void
+getNumOfMCQ():int
+addSubjective(SubName:string):void
+addObjective(ObjNum:int):void
+viewSubjective(SubName:string):void
+viewObjective(ObjName:string):void
Schedule
-day:string
-DateTime:dateTime
-Duration:time
-startingTime:time
-endingTime:time
+setDuration(duration:time)::void
+getDuration():time
+setStartingTime(startingTime:time)::void
+getStartingTime():time
+setEndingTime(endingTime:time)::void
+getEndingTime():time
1..*
1..*
1
1..*
1
1
1
1
1..*
1..*
1
1..*
1..*
1
1..*
1
1..*
1..*
1..*
1..*

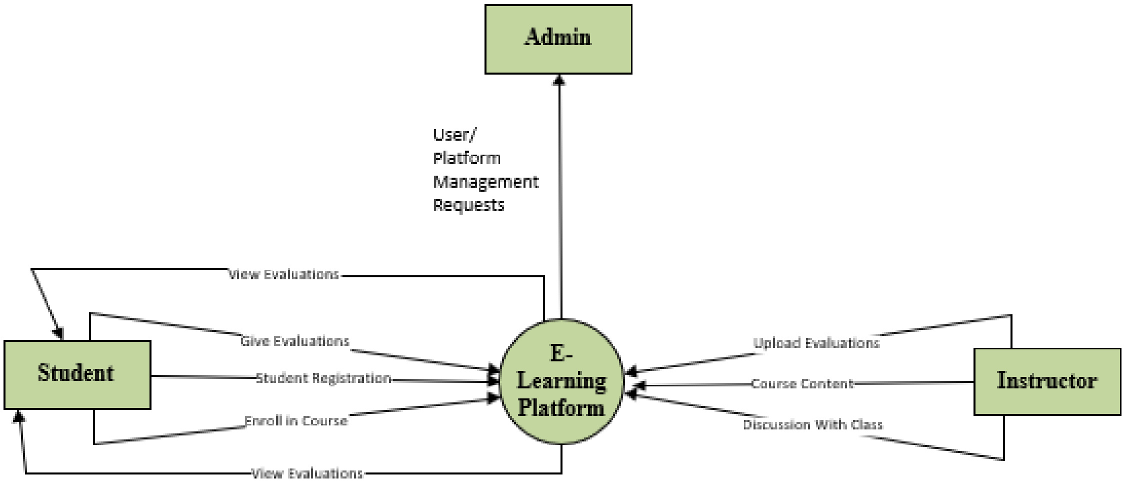
Activity Diagram



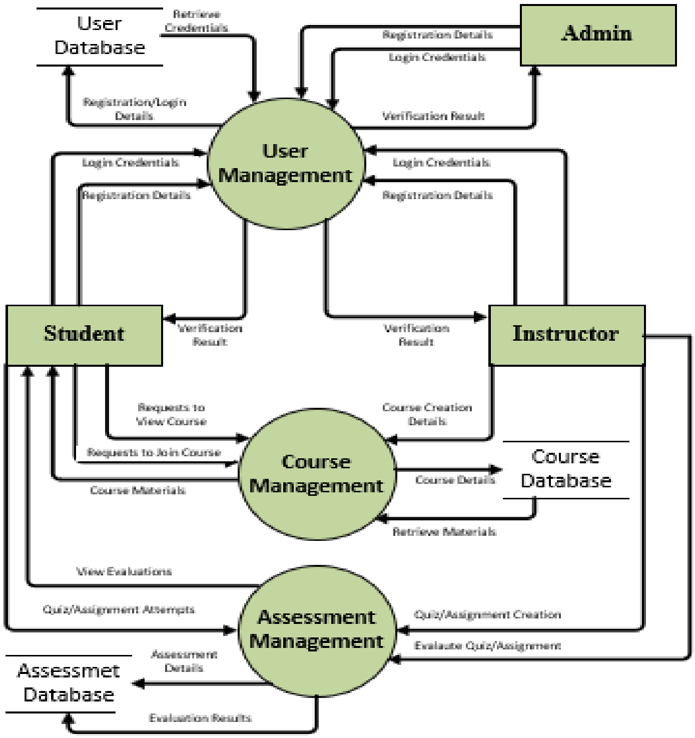
Sequence Diagram



DFD Level 0

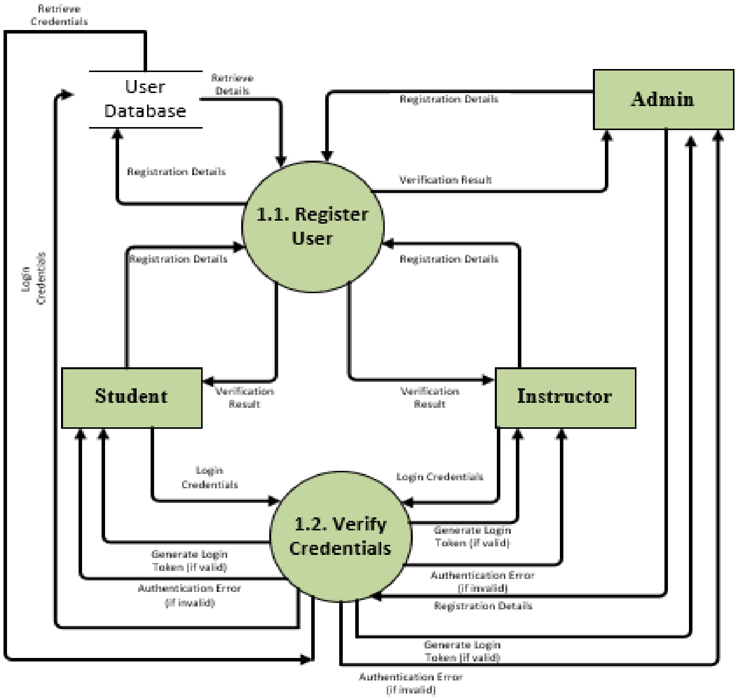


DFD Level 1

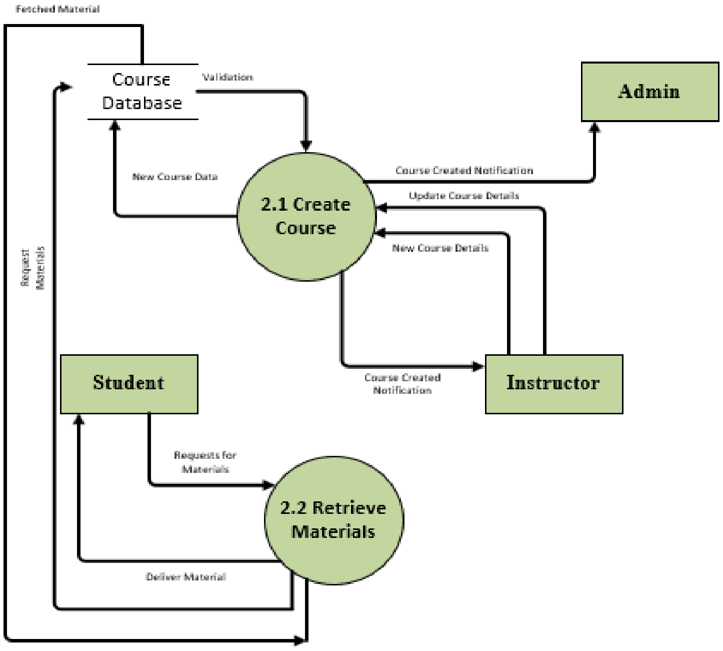


DFD Level 2

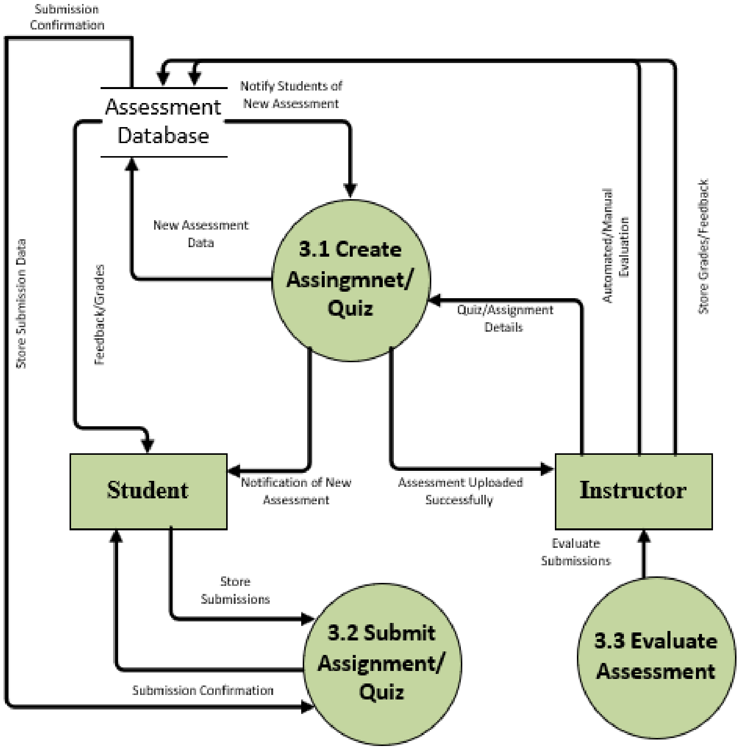
For User Management



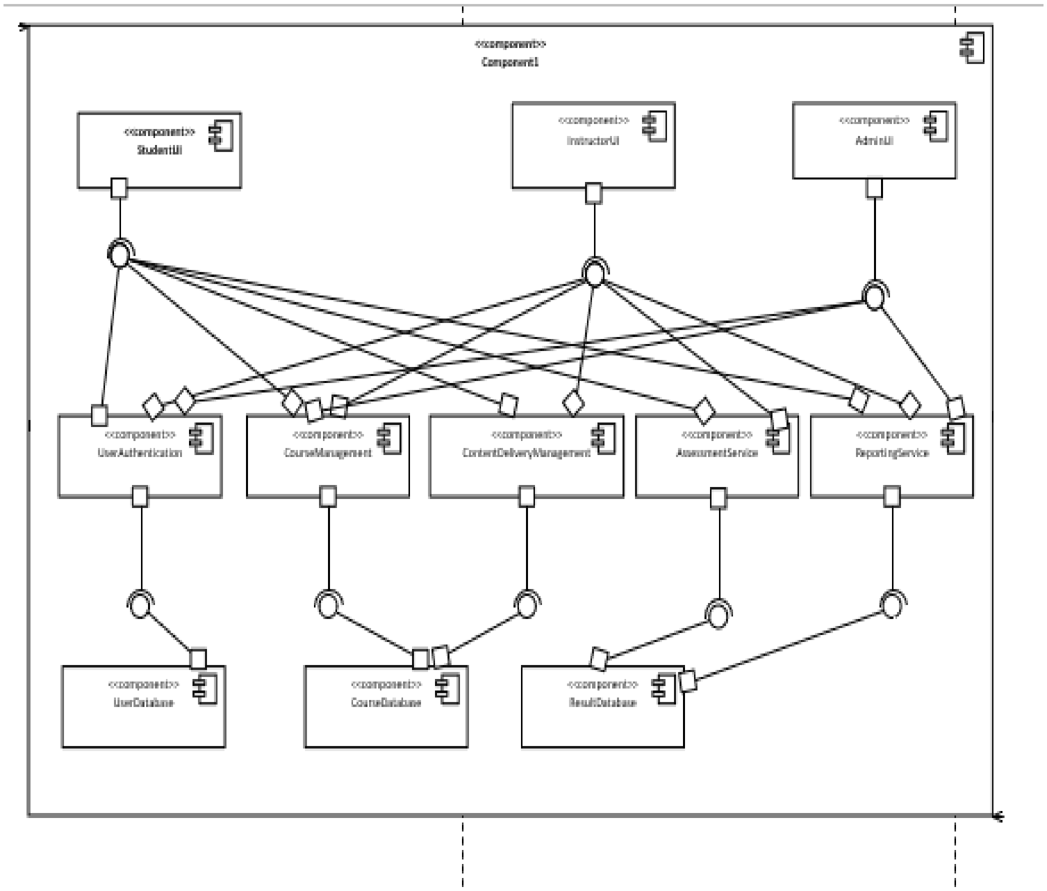
For Course Management



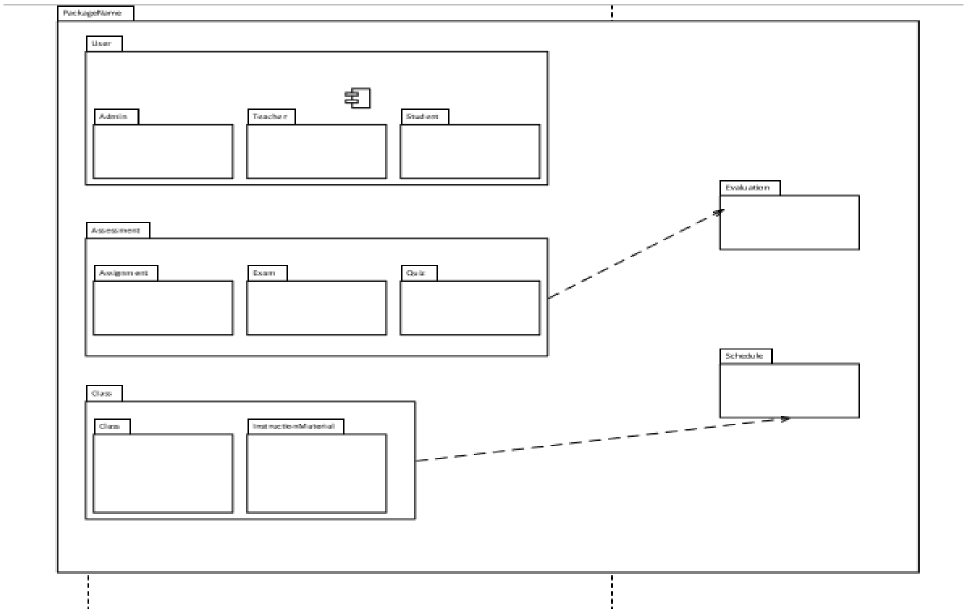
For Assessment Management



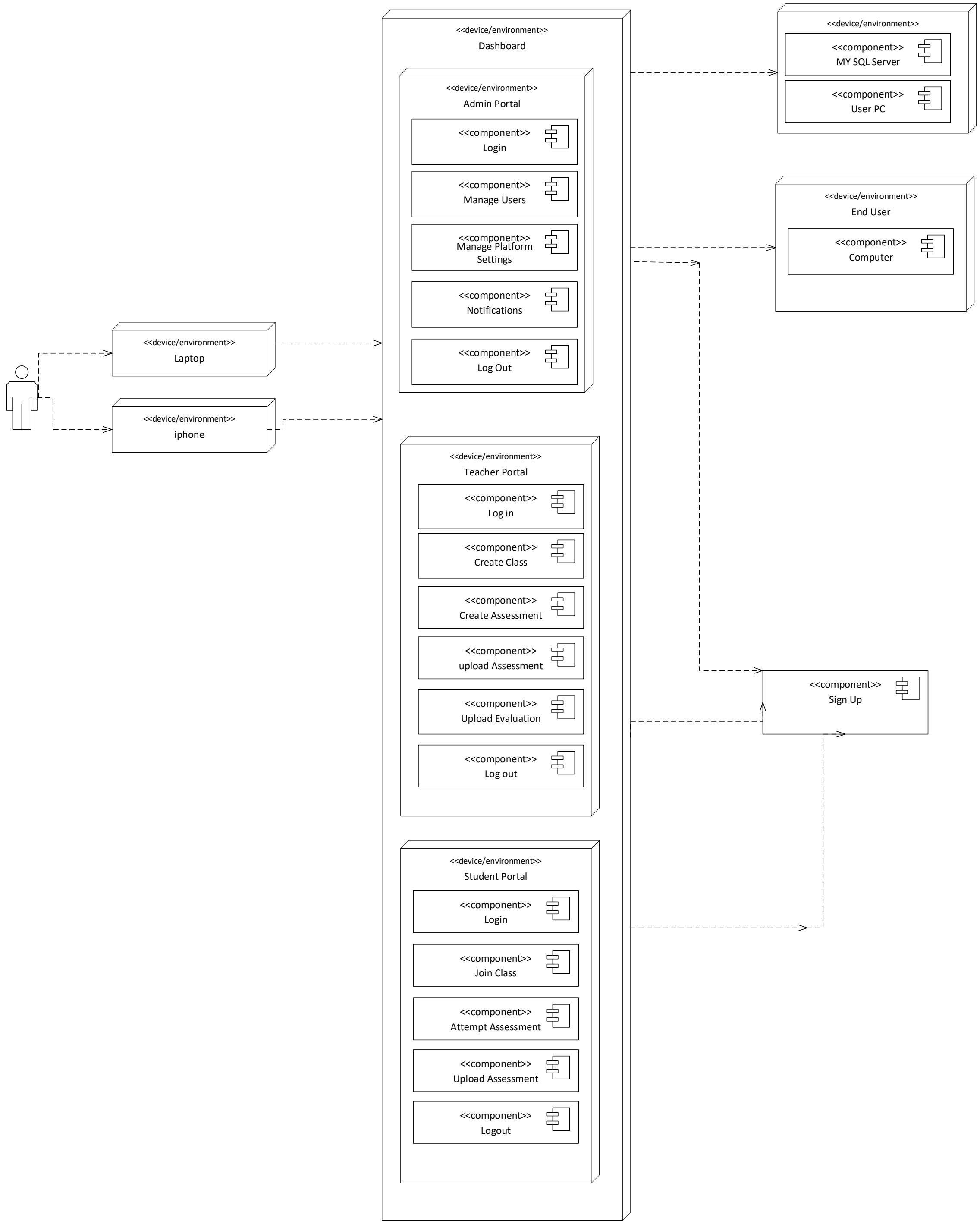
Component Diagram



Package Diagram



Deployment Diagram


<<device/environment>>
Dashboard
<<device/environment>>
Admin Portal
<<component>>
Login
<<component>>
Manage Users
<<component>>
Manage Platform Settings
<<component>>
Notifications
<<component>>
Log Out
<<device/environment>>
Teacher Portal
<<component>>
Log in
<<component>>
Create Class
<<component>>
Create Assessment
<<component>>
upload Assessment
<<component>>
Upload Evaluation
<<component>>
Log out
<<device/environment>>
Student Portal
<<component>>
Login
<<component>>
Join Class
<<component>>
Attempt Assessment
<<component>>
Upload Assessment
<<component>>
Logout
<<device/environment>>
<<component>>
MY SQL Server
<<component>>
User PC
<<device/environment>>
End User
<<component>>
Computer
<<component>>
Sign Up
<<device/environment>>
Laptop
<<device/environment>>
iphone